

REMARKS

Claims 1, 5-11 and 13-16, as amended, remain herein.

Claim 1 has been reworded to recite:

a programming station for generating an automation application, said automation application to be executed in automation equipment and comprising an automation application program written in at least one graphic automation language

See applicants' specification, page 3, second full paragraph.

Claim 1 has been reworded to further recite:

an internal memory for storing a plurality of grammar files written in text format and compliant with eXtensible Markup Language (XML) syntax

said internal memory for storing a plurality of application description files, each said application description files being expressed in XML language and describing part of said automation application,

Serial No. 10/073,193

See applicants' specification, page 3, second full paragraph, and page 6, last paragraph, continuing to page 7. Claim 1 has been reworded to further recite:

said programming station is for using at least one of said grammar files to generate at least one of said application description files describing part of said automation application program,

See applicants' specification, page 3, second and third full paragraphs. Claims 1 and 7 have been reworded to further recite:

wherein said application description files comprise an automation application program description file, an application input-output description file and an application data description file

See now canceled claim 4. Claim 4 has been cancelled without prejudice or disclaimer.

1. The specification has been amended to include headings, thereby mooting the objections thereto.

2. Claims 1 and 4-10 were rejected under 35 U.S.C. §112, first paragraph, as allegedly containing subject matter not described in the specification, i.e., claim 1, lines 5-7, reciting "a plurality of grammar files all written in XML language in text format." The Office Action mentions that the specification, page 14, lines 7-17, describes grammar files as being in either the ".dtd" or ".xsd" formats, and assumes, without sound basis, that such formats are distinct from the XML language, and that "[t]he specification does not disclose that the grammar files may be in any other format (i.e., XML)".

Contrary to the erroneous assumptions in the Office Action, "Attachment A" and "Attachment B" attached hereto demonstrate that it is well known that an ".xsd" file is a file written in XML language and compliant with XML syntax. For instance, ".NET and XML: XSD Schemas" (Attachment A), third section, titled "The Fundamentals of XSD Schemas," explains:

But, very often when processing a XML document, you want to know that it conforms to a certain structure, the structure your application understands. That is where XSD schemas come into play. XSD schemas are the successor of DTDs (Document Type Definition), the difference being that XSD itself uses a XML syntax. XSD schemas allow

Serial No. 10/073,193

you to declare the structure of an XML document, which elements and attributes are allowed, is it mandatory or optional element, can there be more than one instance of an element, and so forth..."

See also, "XML Schema Definition (XSD)" (Attachment B), section titled "What is XML Schema Definition (XSD)," which describes:

XSD: A XML based language that defines validation rules for XML files. XSD stands for XML Schema Definition.

Main features of XSD:

- * XSD is an XML based language, meaning that XSD statements will be written in XML files.

- * XSD defines validation rules for XML files, meaning that XSD can be used to replace DTD, which is another language for defining XML validation rules.

These two references state that "XSD itself uses a XML syntax" and "XSD statements will be written in XML files." Accordingly, the statement in the Office Action that "[t]he specification does not disclose that the grammar files may be in any other format (i.e., XML)" is erroneous because the specification, appendices 4-6, describes examples of ".xsd" files that are written in a text format compliant with XML

Serial No. 10/073,193

syntax, whereas appendices 1-3 show examples of ".dtd" files that are written in a text format not compliant with XML syntax. Thus, the specification discloses a plurality of grammar files written in text format and compliant with XML syntax.

Reconsideration and withdrawal of the rejection are respectfully requested.

3. Claims 1 and 4-10 were rejected under 35 U.S.C. §112, second paragraph. Claim 1 has been reworded and claim 4 canceled, thereby mooting the rejection. Reconsideration and withdrawal of the rejection are respectfully requested.

4. Claims 11 and 13-16 were rejected under 35 U.S.C. §112, second paragraph, as being indefinite. Claims 10, 11 and 13-16 have been reworded, thereby mooting the rejection. Reconsideration and withdrawal of the rejection are respectfully requested.

Serial No. 10/073,193

5. Claims 1, 4, 5, 11, 13, 14 and 16 were rejected under 35 U.S.C. §102(e) over Muenzel Published U.S. Patent Application 2002/004804. Claim 4 has been canceled, thereby mooting its rejection.

The presently claimed programming station is for generating an automation application, the automation application to be executed in automation equipment and including an automation application program written in at least one graphic automation language, the programming station including (1) an internal memory for storing a plurality of grammar files written in text format and compliant with eXtensible Markup Language (XML) syntax, (2) each grammar file including a description grammar describing a syntax of a respective graphic automation language, the internal memory for storing a plurality of application description files, each of the application description files being expressed in an XML language and describing part of the automation application, and the programming station for using at least one of the grammar files to generate at least one of the application description files describing part of the automation application program, wherein the application description files include an automation

Serial No. 10/073,193

application program description file, an application input-output description file and an application data description file. This arrangement and method are nowhere disclosed or suggested in the cited reference.

The Office Action alleges that an automation application including a program description file, an application input-output description file and an application data description file, is "considered inherent to automation control systems; paragraph 003 (of Muenzel '804)." Applicants respectfully disagree.

Automation control systems do not necessarily have to include the three subject files, and there is nothing in the present record that supports the condition that all three must be present. Inherent anticipation requires that the missing descriptive material is "necessarily present," not merely probably or possibly present, in the prior art. In re Robertson, 169 F.3d 743, 745, 49 USPQ2d 1949, 1950-51 (Fed. Cir. 1999) (citing Continental Can Co. USA, Inc. v. Monsanto Co., 948 F.2d 1264, 1268, 20 USPQ2d 1746, 1749 (Fed. Cir. 1991)).

Moreover, Muenzel '804 does not disclose that the internal memory of a programming station stores application description

Serial No. 10/073,193

files including an automation application program description file, an application input-output description file and an application data description file, all written in XML. Instead, Muenzel '804 describes only an "automation application" first written in a graphic automation language, and then converted in an automation application program description file written in XML language. Muenzel '804 does not disclose the initial automation application as a set of automation application description files including an automation application program description file, an application data file, and an application input-output file associated, all described in XML. An entire automation application program usually includes application data and application input-output associated with the automation application program. Applicants' claims 1 and 11 (including former claim 4 subject matter) are directed to all three portions of an automation application program described in XML, (i.e., the automation application program description file, the application input-output description file and the application data file), so that these files can be easily imported from or exported to different third party software (like Electrical CAD

Serial No. 10/073,193

for input-output, SCADA for application data, Microsoft EXCEL, etc.). Thus, the whole automation application program, not just the portion written in a graphical programming language, is described in a standardized XML language. That is, the programming station is for using at least one of the grammar files to generate at least one of the application description files expressed in XML language and describing part of the automation application, wherein the application description files include an automation application program description file, an application input-output description file and an application data description file, as recited in applicants' claims 1 and 11.

For the foregoing reasons, Muenzel '804 fails to disclose, either expressly or inherently, all elements of applicants' claimed invention, and therefore is not a proper basis for rejection under §102. And, there is no disclosure or teaching in Muenzel '804 that would have suggested the desirability of modifying any portions thereof effectively to anticipate or suggest applicants' presently claimed invention. Claims 5-10, which depend from claim 1, are allowable for the same reasons

Serial No. 10/073,193

explained herein for claim 1, and claims 13-16, which depend from claim 11, are allowable for the same reasons explained herein for claim 11. Accordingly, reconsideration and withdrawal of this rejection are respectfully requested.

6. Claims 6-10 were rejected under 35 U.S.C. §103(a) over Muenzel '804 and Nixon et al. U.S. Patent 5,801,942.

Claims 6-10, which depend from claim 1, are allowable for the same reasons explained herein for claim 1.

Moreover, the Office Action admits that Muenzel '804 does not disclose specific links, jumps and coils, and cites Nixon et al. '942 as allegedly teaching same. However, Nixon et al. '942 does not include disclosure that would cure the deficiencies of Muenzel '804, as explained herein.

For the foregoing reasons, neither Muenzel '804 or Nixon et al. '942 contains any teaching, suggestion, reason, motivation or incentive that would have led one of ordinary skill in the art to applicants' claimed invention. Nor is there any disclosure or teaching in either of these references that would have suggested the desirability of combining any portions

Serial No. 10/073,193

thereof effectively to anticipate or suggest applicants' presently claimed invention. Accordingly, reconsideration and withdrawal of this rejection are respectfully requested.

7. Claim 15 is rejected under 35 U.S.C. §103(a) over Muenzel '804 and Lau U.S. Patent 6,598,219.

Claim 15, which depends from claim 11, is allowable for the same reasons explained herein for claim 11.

Moreover, the Office Action admits that Muenzel '804 does not disclose means of checking that the description of the application in the XML language satisfies the description grammar of the graphic automation language used, and cites Lau '219 as allegedly teaching same. However, Lau '219 does not include disclosure that would cure the deficiencies of Muenzel '804 as explained herein.

All claims 1, 5-11 and 13-16 are now proper in form and patentably distinguished over all grounds of rejection stated in the Office Action. Accordingly, allowance of all claims 1, 5-11 and 13-16 is respectfully requested.


Serial No. 10/073,193

Should the Examiner deem that any further action by the applicants would be desirable to place this application in even better condition for issue, the Examiner is requested to telephone applicants' undersigned representatives.

Respectfully submitted,

PARKHURST & WENDEL, L.L.P.

February 14, 2005
Date



Roger W. Parkhurst
Registration No. 25,177
Robert N. Wieland
Registration No. 40,225

RWP:RNW/jmz:klb:mhs

Attachments: (A) ".NET and XML: XSD Schemas: The Fundamentals
of XSD Schemas"
(B) "XML Schema Definition (XSD): What is XML
Schema Definition (XSD)"

Attorney Docket No.: SCHN:018

PARKHURST & WENDEL, L.L.P.
1421 Prince Street, Suite 210
Alexandria, Virginia 22314-2805
Telephone: (703) 739-0220



Download Cloudscape—the embeddable, open source database from IBM

Download Cloudscape

the embeddable, open source database from IBM

Win a 40GB IPOD

IT Management | Networking & Communications | Web Development | Hardware & Systems | Software Development

developer.com®

Search

[go](#)

EARTH

CodeGuru | Gamelan | Jars | Wireless | Discussions

Navigate developer.com

- Architecture & Design
- Database
- Java
- Languages & Tools
- Microsoft & .NET
- Open Source
- Project Management
- Security
- Techniques
- Voice
- Web Services
- Wireless/Mobile
- XML

Compare alternatives for enterprise VoIP and VoIP access. Find out how to make the best VoIP business decision for your company.

.NET and XML: XSD Schemas

By Klaus Salchner

W3C (the World Wide Web Consortium, <http://www.w3.org>) published the XML 1.0 specification on February 10th, 1998. The XML 1.1 specification was published six years later, on February 4th 2004. In the six years, XML has taken the industry by storm. XML has become the standard for how to describe and exchange data. The current development platforms, .NET and J2EE, support XML natively. All modern enterprise applications, be it a SQL Server or Oracle database, a BizTalk Server, an Office suite, or any of the other thousands of applications support XML to various degrees. You will be pretty hard pressed to find an application that does not support or use XML.

The first article explained the fundamentals and powers of XPath queries. XPath queries allow you to search and navigate your XML documents easily. This article looks at the fundamentals and powers of XSD schemas. The following articles look at XSL Transformations and then how well these three standards are supported by the .NET framework and what the most important namespaces and types are. This series of articles is not intended as a comprehensive description of all the .NET types around XML. The goal is rather provide a good introduction so you understand the XML capabilities of the .NET framework and leveraging them for your current .NET projects.

The Sample XML Document for the Series of Articles

This series of articles takes it as a given that you are familiar with XML itself. The sample XML

Microsoft

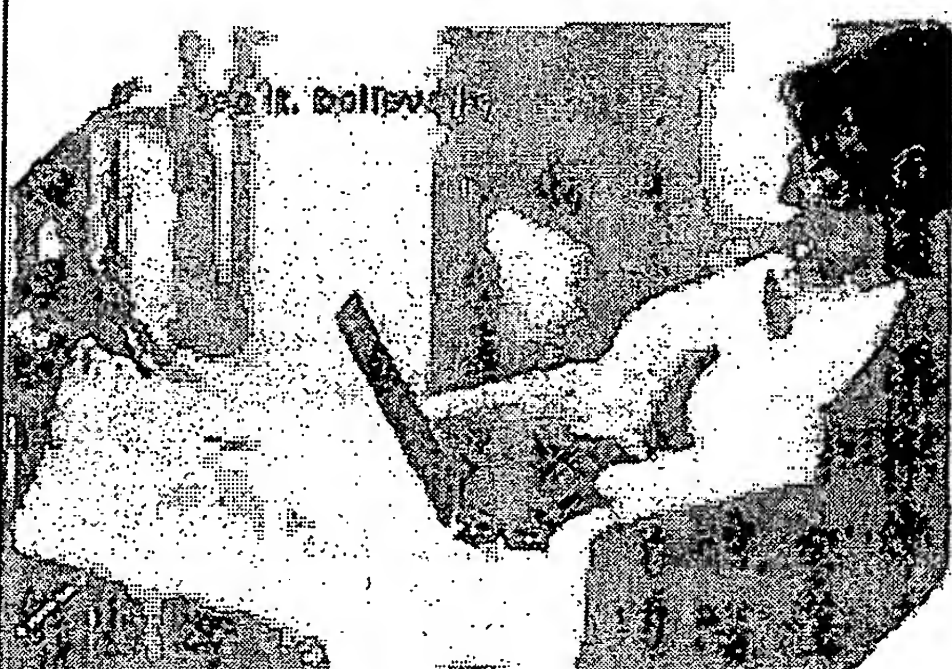
.net

Take Advantage of the Full .NET Framework with VS '05 and SQL Server '05

AZ, Phoenix	Mar 4
CA, Los Angeles	Feb 9
CA, San Diego	Feb 8
CA, San Francisco	Feb 11
CO, Denver	Mar 2
DC, Washington	Feb 17
FL, Fort Lauderdale	Feb 18
GA, Atlanta	Feb 16
IL, Chicago	Feb 14

Royalty free stock photography by subscription.

PAY ONE FEE. DOWNLOAD WHAT YOU NEED.



Developer News

- [Apple's iPod Shuffle Stifles Podcasting](#) Janua 2005
- [LeapFrog Launches Developer Network](#) Janu 2005
- [Torvalds Criticizes Security Approaches](#) Janu 2005
- [Linux Heavies Issue Patches](#) January 13, 2005

Free Tech Newsletter

Enterprise Networking Planet ☒

Your E-mail

[Sign Up](#)

MA, Boston	Feb 15
MI, Detroit	Feb 7
MN, Minneapolis	Mar 2
NY, New York	Feb 14
OH, Cincinnati	Feb 9
OR, Portland	Mar 4
PA, Philadelphia	Mar 15
TN, Nashville	Mar 9
TX, Dallas	Feb 28
WA, Seattle	Feb 11
Click here for all cities	



[Be a Commerce Partner](#)
[UK Shopping](#)
[Web Hosting Search](#)
[Domain Registration](#)
[Technology Job Search](#)
[Cheap Flights](#)
[Dedicated Servers](#)
[Phone Systems](#)
[Web Hosting Services](#)
[Register Domain Name](#)
[Internet Marketing](#)
[Online College](#)
[Boat Donations](#)



Compare products, prices, and stores at Hardware Central!

Computers
[Desktops,](#)
[Mac & PC Notebooks,](#)
[Monitors, Scanners,](#)
[Webcams, PDA's,](#)
[more...](#)

used throughout the articles is a list of employees, which must have for each employee the first name, phone number, and e-mail address and can also provide the job title and a Web address

```

<?xml version="1.0" encoding="utf-8"?>
<Employees xmlns="http://tempuri.org/MySchema.xsd">
  <Employee ID="1">
    <FirstName>Klaus</FirstName>
    <LastName>Salchner</LastName>
    <PhoneNumber>410-727-5112</PhoneNumber>
    <EmailAddress>klaus_salchner@hotmail.com</EmailAddress>
    <WebAddress>http://www.enterprise-minds.com</WebAddress>
    <JobTitle>Sr. Enterprise Architect</JobTitle>
  </Employee>
  <Employee ID="2">
    <FirstName>Peter</FirstName>
    <LastName>Pan</LastName>
    <PhoneNumber>604-111-1111</PhoneNumber>
    <EmailAddress>peter.pan@fiction.com</EmailAddress>
    <JobTitle>Sr. Developer</JobTitle>
  </Employee>
</Employees>

```

The Fundamentals of XSD Schemas

It is very easy to create XML documents whether programmatically or manually through an XML Spy, Stylus Studio, or Visual Studio .NET 2003. But, very often when processing a XML document, you want to know that it conforms to a certain structure, the structure your application understands where XSD schemas come into play. XSD schemas are the successor of DTDs (Document Type Definitions), the difference being that XSD itself uses a XML syntax. XSD schemas allow you to declare the structure of an XML document, which elements and attributes are allowed, is it a mandatory or optional element, can there be more than one instance of an element, and so forth. You then can use the XSD schema to validate the XML document, meaning does the XML document conform to the structure described by the schema. The XML describes the data and the XSD schema describes the structure of the data. The first version of the XSD schema standard has been released May 2001 and can be found at <http://www.w3.org/TR/xmlschema-0/>, <http://www.w3.org/TR/xmlschema-1/> along with <http://www.w3.org/TR/xmlschema-2/>. The working draft of XSD 1.1 can be found at <http://www.w3.org/TR/2003/WD-xmlschema-11-req-20030121/>.

When you create your XSD schema, you do two things. First, you declare an element or attribute. Declaring means you associate an element or attribute name with a set of constraints, for example, an element with the name FirstName is of the string type and only one element of that name is allowed. Second, you define new simple or complex types. XSD has a set of standard types such as string, integer, date, and so forth. The .NET framework maps these XSD data types against its .NET classes. In our sample XML document, the Employee is a complex type. Think in terms of data structures: in application code, you would define a new structure called Employee and it would contain the elements FirstName, LastName, PhoneNumber, EmailAddress, WebAddress, and JobTitle. In XSD schema, you define exactly the same. You define a complex type of the name Employee and then declare all the elements it has plus the constraints for each element; for example, the FirstName element is of the string type. See the XSD below schema for our sample XML document:

```

<?xml version="1.0"?>
<xs:schema targetNamespace="http://tempuri.org/MySchema.xsd"
  xmlns="http://tempuri.org/MySchema.xsd"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  attributeFormDefault="unqualified"
  elementFormDefault="unqualified">
  <xs:element name="Employees">
    <xs:complexType>
      <xs:choice minOccurs="1" maxOccurs="unbounded">
        <xs:element name="Employee" type="EmployeeType"/>
      </xs:choice>
    </xs:complexType>

```

XML Schema Definition (XSD)

What is XML Schema Definition (XSD)

XSD: A XML based language that defines validation rules for XML files. XSD stands for XML Schema Definition.

Main features of XSD:

- XSD is an XML based language, meaning than XSD statements will be written in XML files.
- XSD defines validation rules for XML files, meaning that XSD can be used to replace DTD, which is another language for defining XML validation rules.

"Hello world!" Example of XSD

Let's write our first XSD file to define the structure for our "Hello world!" XML file. Here is the XSD file, hello.xsd:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="p" type="xsd:string"/>
</xsd:schema>
```

The XSD file needs to be linked to the XML file, hello_xsd.xml:

```
<?xml version="1.0"?>
<p xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="hello.xsd">
Hello world!</p>
```

Note that:

- The XSD file is linked to the XML file through an attribute in the root element.

Validating XML Files Against XSD Files

Not all the XML file browsers support XSD validation. For example, Microsoft Internet Explorer 6.0 browse XML files every well as I mentioned in my previous notes, but I don't know how to make it to validate XML files against the linked XSD files.

XSD validation tools:

- XML Spy 5.3